A data-driven shock capturing approach for discontinuous Galekin methods

Jian Yu[a], Jan S Hesthaven[b], Chao Yan[a,*]

[a]School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China
[b]Chair of Computational Mathematics and Simulation Schience, École Polytechnique Fédérale de Lausanne, Lausanne CH-1015, Switzerland

Abstract

We propose a data-driven artificial viscosity model for shock capturing in discontinuous Galerkin methods. The proposed model trains a multi-layer feedforward network to map from the element-wise solution to a smoothness indicator, based on which the artificial viscosity is computed. The data set for the training of the network is obtained using canonical functions. The compactness of the data set, which is critical to the success of training the network, is ensured by normalization and the adjustment of the range of the smoothness indicator. The network is able to recover the expected smoothness much more reliably than the original averaged modal decay model. Several smooth and non-smooth test cases are considered to investigate the performance of this approach. Convergence tests show that the proposed model recovers the accuracy of the corresponding linear schemes for smooth regions. For non-smooth flows, the model is observed to suppress spurious oscillations well.

Keywords: Discontinuous Galerkin, Artificial viscosity, Artificial neural network

1. Introduction

In recent years, there has been an increasing demand for high-fidelity simulations of compressible flows, which poses great challenges for numerical methods. High order methods, including finite difference methods, finite volume methods, and spectral element methods are considered to be effective approaches for such purposes. Despite their good accuracy and efficiency, finite difference methods require block-structured grids of high quality, which is challenging for complex configurations [1]. Finite volume methods are suitable for unstructured grids, but result in extensive stencils to achieve high orders [2]. Spectral element methods, such as the discontinuous Galerkin (DG) method [3], the flux reconstruction method [4], and the spectral volume/difference methods [5], are intrinsically designed to achieve high order accuracy on unstructured grids. Among these methods, DG has drawn great attention [6], and still undergoes rapid development.

One major challenge when simulating strongly compressible flows with DG is shock capturing. Available methods on this topic [7] include slope limiters, WENO (weighted essentially nonoscillatory) limiters, and artificial viscosity models. With some indicator, slope limiters first identify troubled cells on which the linear mode is limited, while higher order components are discarded [3]. Such limiters are able to ensure the total variation bounded(TVB) property, but often result in a detrimental effect on the high order accuracy [8, 9]. WENO limiters also rely on reliable troubled-cell indicators [10], and WENO reconstruction for either point-wise values or modal coefficients in troubled cells. In the early work of such methods, only cell averages were used for reconstruction, and a large stencil is required for high orders [11, 12]. To overcome this issue, a family of Hermite WENO limiters [13–15] was developed, which use derivatives besides cell averages for reconstruction, resulting in a reduced stencil size. The key for WENO/HWENO limiters is to involve as much available information as possible to derive a compact limiter. Following this line, [16, 17] proposed to reconstruct the entire polynomials, i.e. the approximation polynomial on each troubled cell as well as its immediate neighbors are combined to define the new WENO reconstruction polynomial on the troubled cell. This method requires a stencil of the von Neumann neighborhood, and delivers good performance for low order

*Corresponding author
Email addresses: yuj@buaa.edu.cn (Jian Yu), Jan.Hesthaven@epfl.ch (Jan S Hesthaven), yanchao@buaa.edu.cn (Chao Yan)

accuracy. However, it is difficult to extend such WENO limiters to higher order cases (typically higher than 3rd order) due to the stability issue of extrapolation.

A third strategy for shock capturing is to add an explicit dissipation term to the original system, relying on some smoothness sensor. In the context of finite difference methods, [18, 19] initiated an approach which employs high order derivatives of dilation to stabilize shocks, followed by a series of improvements [20–22]. [23] extends this idea to the spectral volume method, and [24] further enhances the method by scaling the dilation. It is worth noting that computation of higher order derivatives generally involves considerable complexity and cost especially for unstructured grids. Another approach is to use the entropy production to determine the amount of the artificial viscosity, and is implemented in DG [25, 26]. [27] further examines how to reduce dissipation of the entropy method in viscous regions, making the method applicable to viscous cases. As a third approach, [28] proposes to use the decay rate of the modal coefficients to measure smoothness originating in the assumption that in an approximate expansion for a continuous function, the P-th mode scales as $1/P^2$. Despite its reasonable success, this method is often based on the highest mode, providing limited information on the decay rate, and tends to underestimate the smoothness. Therefore, [29] proposed to involve all the modes (except the first mode which denotes cell average) to estimate the decay rate with a least-square method.

In this work, we propose a data-driven artificial viscosity approach, which takes the decay rate as the smoothness indicator as well. However, instead of computing the decay rate based on the modal coefficients, we use an artificial neural network (ANN) to build a map from the solution to the decay rate. There have been various successful applications of machine learning techniques in the CFD community, mainly on turbulence modeling. [30] examines how well a trained neural network can reproduce the source term of the Spalart-Allmaras model, and [31] proposes a tensor basis neural network, which embeds Galilean invariance and is trained to predict the Reynolds stress anisotropy. A transition model for a bypass transition boundary layer flow was effectively enhanced with machine learning techniques in [32]. More work along this line can be found in [33], and the references therein. Recently, the limitations of the black-box approach (i.e. treating the entire numerical model as a black-box) have been recognized [34–38], and the advantages of exploiting the prior information about the model have been illustrated. In light of this, we follow the open-box approach, and attempt to derive a shock capturing method based on artificial neural networks.

The paper is organized in the following manner. Section 2 describes the numerical discretization methods and the governing equations used in this paper. Section 3 briefly introduces the mathematical principles of a feedforward ANN. Section 4 presents the proposed data-driven shock capturing approach and Section 5 shows several benchmarks to test the performance of the propose method, followed by some conclusions in Section 6.

2. Numerical discretization

2.1. Governing equation

A convection-diffusion system is considered, i.e.

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{f} - \nabla \cdot \boldsymbol{g} = 0, \tag{2.1}$$

where $\boldsymbol{q}$ is the conserved variables, $\boldsymbol{f}$ is the convective flux, and $\boldsymbol{g}$ is the viscous flux. In this work, $\boldsymbol{g}$ provides artificial diffusion for the purpose of stabilizing shocks, and takes the Laplacian form, given as

$$\boldsymbol{g} = \mu \boldsymbol{w}, \quad \boldsymbol{w} = \nabla \boldsymbol{q}, \tag{2.2}$$

where μ is an artificial viscosity, which will be described in more detail below.

2.2. Nodal discontinuous Galerkin formulation

The nodal discontinuous Galerkin method, developed in [39], is used in this work. For simplicity, the scalar case will be considered in the following, but the extension to systems is straightforward.

2.2.1. One-dimensional case

The standard element I^{1D} is defined as $I^{1D} = [-1, 1]$, and an affine mapping is introduced

$$x(r) = x_{v_0} + \frac{1+r}{2}h, \quad h = x_{v_1} - x_{v_0}, \quad r \in [-1, 1]. \tag{2.3}$$

Here the subscripts v_0 and v_1 represent the left and right end of the element, respectively.

The nodal expression of the approximate solution $q^h(r, t)$ on I^{1D} is given as

$$q^h(r, t) = \sum_{m=0}^{N^P-1} \tilde{q}_m(t) l_m(r), \quad \tilde{q}_m(t) \equiv q^h(r_m, t), \tag{2.4}$$

where $N^P = P + 1$, P is the order of the polynomial, $\tilde{q}_m(t)$ is the nodal value at $r = r_m$, and $l_m(r)$ is the Lagrange polynomial of Pth order. The nodes on which the Lagrange polynomials are based are chosen to be the Legendre-Gauss-Lobatto points.

Similarly, the modal formulation for $q^h(r, t)$ can be written as

$$q^h(r, t) = \sum_{m=0}^{N^P-1} \hat{q}_m \varphi_m(r), \tag{2.5}$$

where $\varphi_m(r)$ belongs to an orthonormal basis, and is chosen to be

$$\varphi_m(r) = \frac{Q_m^{(0,0)}(r)}{\sqrt{\chi_m}}, \qquad \chi_m = \frac{2}{2m+1}, \tag{2.6}$$

where $Q_m^{(\alpha,\beta)}(r)$ is the Jacobi polynomial of order m, and χ_m is the normalization term. Note that for $\alpha = 0, \beta = 0$, $Q_m^{(\alpha,\beta)}(r)$ reduces to the Legendre polynomial.

The relation between the nodal and modal formulations can be established with the Vandermonde matrix $\boldsymbol{V}$ as

$$\tilde{\boldsymbol{q}} = \boldsymbol{V}\hat{\boldsymbol{q}}, \quad V_{m,n} = \varphi_n(r_m), \quad (m, n) \in [0, N_P - 1]. \tag{2.7}$$

Similar to q, the nodal approximations for the fluxes in (2.1) and (2.2) can be expanded as

$$f^h(r, t) = \sum_{m=0}^{N_P-1} \tilde{f}_m(t) l_m(r), \quad \tilde{f}_m(t) \equiv f(q^h(r_m, t)), \tag{2.8}$$

$$g^h(r, t) = \sum_{m=0}^{N_P-1} \tilde{g}_m(t) l_m(r), \quad \tilde{g}_m(t) \equiv g(q^h(r_m, t)), \tag{2.9}$$

$$w^h(r, t) = \sum_{m=0}^{N_P-1} \tilde{w}_m(t) l_m(r), \quad \tilde{w}_m(t) \equiv w(q^h(r_m, t)). \tag{2.10}$$

Substituting (2.8) - (2.10) into (2.1) - (2.2), we recover

$$R_1^h = \frac{\partial q^h}{\partial t} + \frac{\partial f^h}{\partial x} - \frac{\partial g^h}{\partial x}, \quad R_2^h = w^h - \frac{\partial q^h}{\partial x}. \tag{2.11}$$

The Galerkin method requires that

$$\int_K R_1^h l_n dK = 0, \quad \int_K R_2^h l_n dK = 0, \quad n = 0, ..., N_P - 1. \tag{2.12}$$

The weak form of the nodal DG discretization is obtained from (2.12) after integration by parts

$$\frac{h}{2}\boldsymbol{M}\frac{d\tilde{\boldsymbol{q}}}{dt} - \boldsymbol{S}^T\tilde{\boldsymbol{f}} + \boldsymbol{S}^T\tilde{\boldsymbol{g}} = -\boldsymbol{I}_P f^*_{v_1} + \boldsymbol{I}_0 f^*_{v_0} + \boldsymbol{I}_P g^*_{v_1} - \boldsymbol{I}_0 g^*_{v_0}, \tag{2.13}$$

$$\frac{h}{2}\boldsymbol{M}\tilde{\boldsymbol{w}} + \boldsymbol{S}^T\tilde{\boldsymbol{q}} = \boldsymbol{I}_P w^*_{v_1} - \boldsymbol{I}_0 w^*_{v_0}, \tag{2.14}$$

where I_m is a unit vector of $P+1$ entries with the mth entry being 1. f^*, g^* and w^* denote the numerical fluxes at the element interface. For g^* and q^*, the centered flux is employed, while for f^* the Lax-Friedrich scheme is used, i.e.

$$f^* = \{\{f_h\}\} + \frac{\lambda}{2}[\![u]\!], \tag{2.15}$$

where λ is the locally maximum wave speed. $\{\{\bullet\}\}$ and $[\![\bullet]\!]$ denote the algebraic average and the jump at the interface, respectively. $\boldsymbol{M}$ and $\boldsymbol{S}$ are known as the mass and stiffness matrices, respectively, with entries defined as

$$M_{n,m} = \int_{-1}^{1} l_n(r) l_m(r) dr, \quad S_{n,m} = \int_{-1}^{1} l_n(r) \frac{dl_m(r)}{dr} dr. \tag{2.16}$$

From (2.7), we further obtain

$$\boldsymbol{M} = \left(\boldsymbol{V}\boldsymbol{V}^T\right)^{-1}, \quad \boldsymbol{S} = \boldsymbol{M}\boldsymbol{D}^r, \tag{2.17}$$

$$\boldsymbol{D}^r = \boldsymbol{V}^r\boldsymbol{V}^{-1}, \quad V^r_{n,m} = \frac{d\varphi_m}{dr}\Big|_{r_n}. \tag{2.18}$$

2.2.2. Two-dimensional case

The affine mapping between an arbitrary triangular element and its standard counterpart (see Fig. 1) is given as

$$\boldsymbol{x}(r,s) = -\frac{r+s}{2}\boldsymbol{v}_0 + \frac{r+1}{2}\boldsymbol{v}_1 + \frac{s+1}{2}\boldsymbol{v}_2, \tag{2.19}$$

where $\boldsymbol{v}_0$, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are the position vectors corresponding to the three vertices, respectively.

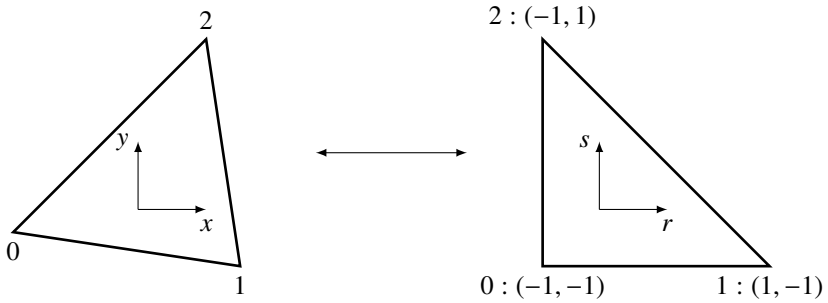


Fig. 1. The affine mapping of a triangular element

The nodal representation on the standard element is given as

$$q^h(r,s,t) = \sum_{m=0}^{N^P-1} \tilde{q}_m(t) l_m(r,s), \quad \tilde{q}_m(t) \equiv q^h(r_m, s_m, t), \tag{2.20}$$

where $N^P = (P+1)(P+2)/2$. The nodal points (r_m, s_m) are chosen to be the α-optimized nodal set [39], which reduces to the one-dimensional Legendre-Gauss-Lobatto points along each edge of the triangular element.

The modal representation for 2D is given as

$$q^h(r,s,t) = \sum_{m=0}^{N^P-1} \hat{q}_m(t)\varphi_m(r,s), \tag{2.21}$$

where the orthonormal basis function $\varphi_m(r,s)$ takes the form

$$\varphi_m(r,s) = \sqrt{2}Q_i^{(0,0)}(a)Q_j^{(2i+1,0)}(b)(1-b)^i, \quad a = 2\frac{1+r}{1-s} - 1, \quad b = s. \tag{2.22}$$

Here (a,b) denotes a collapsed coordinate of a rectangular element transformed from the standard triangle. The index m in $\varphi_m(r,s)$ is related to (i,j) as

$$m = j + (P+1)i - \frac{1}{2}i(i-1), \quad (i,j) \geq 0, \quad i+j \leq P. \tag{2.23}$$

Using the same approach as in 1D, the weak form of the two-dimensional nodal DG formulation is obtained as

$$J\boldsymbol{M}\frac{d\tilde{\boldsymbol{q}}}{dt} - (\boldsymbol{S}^x)^T\tilde{\boldsymbol{f}}^x - (\boldsymbol{S}^y)^T\tilde{\boldsymbol{f}}^y + (\boldsymbol{S}^x)^T\tilde{\boldsymbol{g}}^x + (\boldsymbol{S}^y)^T\tilde{\boldsymbol{g}}^y = -\sum_e J_e^\sigma \boldsymbol{M}_e^\sigma \boldsymbol{n}_e \cdot \boldsymbol{f}^* + \sum_e J_e^\sigma \boldsymbol{M}_e^\sigma \boldsymbol{n}_e \cdot \boldsymbol{g}^*, \tag{2.24}$$

$$J\boldsymbol{M}\tilde{\boldsymbol{w}} + (\boldsymbol{S}^x)^T\tilde{\boldsymbol{q}}\vec{\boldsymbol{i}} + (\boldsymbol{S}^y)^T\tilde{\boldsymbol{q}}\vec{\boldsymbol{j}} = \sum_e J_e^\sigma \boldsymbol{M}_e^\sigma \boldsymbol{n}_e \boldsymbol{q}^*, \tag{2.25}$$

where J is the transformation Jacobian obtained from (2.8), and is constant for straight-sided triangular elements. J_e^σ and $\boldsymbol{M}_e^\sigma$ correspond to the transformation Jacobian and mass matrix along the eth edge, respectively. $\boldsymbol{M}_e^\sigma$ is a $N_p \times (P+1)$ matrix, where $M^\sigma_{e,(m,n)} = \int_{\sigma_e} l_m l_n d\sigma$, with σ_e indicating the eth edge. Note that l_m is equal to zero on points which do not reside on the e-th edge. Consequently, instead of being a full marix, $\boldsymbol{M}_e^\sigma$ only has nonzero entries in those rows, m, where $\boldsymbol{x}_m$ resides on the edge.While the mass matrix $\boldsymbol{M}$ is the same as in 1D, the stiffness matrices are slightly different, defined as

$$\boldsymbol{S}^x = \frac{\partial r}{\partial x}\boldsymbol{S}^r + \frac{\partial s}{\partial x}\boldsymbol{S}^s, \quad \boldsymbol{S}^y = \frac{\partial r}{\partial y}\boldsymbol{S}^r + \frac{\partial s}{\partial y}\boldsymbol{S}^s. \tag{2.26}$$

In practice, we have

$$\boldsymbol{S}^r = \boldsymbol{M}\boldsymbol{D}^r, \quad \boldsymbol{S}^s = \boldsymbol{M}\boldsymbol{D}^s, \tag{2.27}$$

$$\boldsymbol{D}^r = \boldsymbol{V}^r\boldsymbol{V}^{-1}, \quad \boldsymbol{D}^s = \boldsymbol{V}^s\boldsymbol{V}^{-1}, \tag{2.28}$$

$$V^r_{m,n} = \frac{\partial \varphi_n}{\partial r}|_{r_m,s_m}, \quad V^s_{m,n} = \frac{\partial \varphi_n}{\partial s}|_{r_m,s_m}. \tag{2.29}$$

3. Artificial neural networks

As one of the most popular strategies of machine learning, feedforward ANNs are loosely inspired by biological neural networks and are able to learn from observational data. In practice, ANNs have been widely used in a wide range of application fields, including image recognition, speech recognition, and natural language processing. The architecture of an ANN is variable and task-dependent. In this work, multi-layer feedforward ANNs, known to be suitable for regression problems, are chosen, and we will describe them in the following.

3.1. The general setup of a multi-layer feedforward ANN

An ANN is a network of artificial neurons, serving as simple processing elements. In general, neurons are organized into layers, which in turn form the network. A typical ANN is illustrated in Fig. 2. The leftmost layer is the input layer, which takes the input of the ANN, while the rightmost is the output layer, producing the output of the ANN. Between the input and output layers are hidden layers. The term *feedforward* indicates that information flows from left to right in a one-way manner [40]. In Fig. 2, v_m^I, $v_m^{H_l}$ and v_m^O denote the output from neuron m at the input layer, the lth hidden layer and the output layer, respectively. In this work, the ANN is used as a map for $\boldsymbol{X} \in \mathbb{R}^{N^I} \rightarrow \boldsymbol{Y} \in \mathbb{R}^{N^O}$, where N^I and N^O denote the number of neurons in the input and output layers, respectively. The mathematical formulation of a general multi-layer feedforward ANN can be expressed as

$$\begin{cases} \boldsymbol{\nu}^{I} &= \boldsymbol{X}, \\ \boldsymbol{\nu}^{H_1} &= \sigma^{H_1}\left(\boldsymbol{W}^{H_1}\boldsymbol{\nu}^{I} + \boldsymbol{B}^{H_1}\right), \\ \boldsymbol{\nu}^{H_l} &= \sigma^{H_l}\left(\boldsymbol{W}^{H_l}\boldsymbol{\nu}^{H_{l-1}} + \boldsymbol{B}^{H_l}\right), \quad \text{for } l = 2, 3, \ldots, N^L, \\ \boldsymbol{\nu}^{O} &= \sigma^{O}\left(\boldsymbol{W}^{O}\boldsymbol{\nu}^{N^L} + \boldsymbol{B}^{O}\right), \\ \hat{\boldsymbol{Y}} &= \boldsymbol{\nu}^{O}, \end{cases} \tag{3.1}$$

where $\boldsymbol{W}$, $\boldsymbol{B}$ and σ represent the weight matrix, the bias vector, and the activation function, respectively, of the corresponding layer (indicated by the superscript). Supposing N^{H_l} denotes the number of neurons in the lth hidden layer, we have $\boldsymbol{W}^{H_1} \in \mathbb{R}^{N^{H_1} \times N^I}$, $\boldsymbol{W}^{H_l}|_{l>1} \in \mathbb{R}^{N^{H_l} \times N^{H_{l-1}}}$, $\boldsymbol{W}^O \in \mathbb{R}^{N^O \times N^{H_{N^L}}}$, $\boldsymbol{B}^{H_l} \in \mathbb{R}^{N^{H_l}}$, $\boldsymbol{B}^O \in \mathbb{R}^{N^O}$.

From (3.1), it is clear that the weighted sum of the outputs of all the neurons from the previous layer is first added to a bias vector, and then fed into the activation function to generate the output of the current neuron. The term *activation function* originates from its biological counterpart, which activates the neuron whenever the input exceeds some threshold. Note that the activation function σ applies in a component-wise manner. In the literature, there have been various candidates for activation function, including Sigmoid, Tanh, Softsign, and ReLU (rectified linear unit). Among all, ReLU, which computes the function $\sigma(z) = max(0, z)$, has become quite popular for the last few years due to its simplicity and effectiveness. On the other hand, ReLU can be fragile and result in dead neurons, i.e. neurons that never activates again. A number of changes on ReLU have been proposed [41], such as leaky ReLU, parametric ReLU(PReLU), randomized leaky ReLU(RReLU), and exponential linear unit(ELU). However, none has been able to replace ReLU due to inconsistent performance improvements across different datasets [41]. Therefore, ReLU is chosen in our final ANN model.

Besides the activation function, the number of hidden layers N^L and the number of neurons N^{H_l} at each hidden layer remain to be determined. Note that N^I and N^O are determined by the specific problem. For N^L, it has been shown that a multi-layer feedforward ANN with two hidden layers can approximate any function [42, 43], but no clear rules are given to choose N^{H_l}, the determination of which thus relies on a trial-and-error approach in practice.

3.2. Training ANNs

After the architecture of an ANN is decided, the weights $\boldsymbol{W}$ and biases $\boldsymbol{B}$ in (3.1) are the unknown variables and need to be computed through a training process. To train an ANN, a number of examples need to be provided, which are formed into a design matrix $\boldsymbol{\Xi} \in \mathbb{R}^{N^S \times N^I}$, each row corresponding to an example, i.e.

$$\boldsymbol{\Xi}_s = \begin{bmatrix} \boldsymbol{X}^s \\ \boldsymbol{Y}^s \end{bmatrix}^T, \quad \text{for } s = 1, \ldots, N^S. \tag{3.2}$$

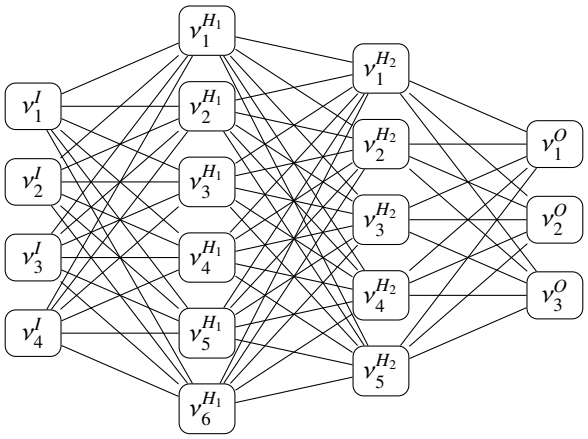


Fig. 2. The architecture of a typical ANN

Note that the ultimate goal of machine learning algorithms is to perform well on previously unseen data. Therefore, it is a common strategy to divide the design matrix into three parts: the training set, the test set, and the validation set, i.e.

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\Xi}^{\text{train}} \\ \boldsymbol{\Xi}^{\text{test}} \\ \boldsymbol{\Xi}^{\text{validation}} \end{bmatrix}. \tag{3.3}$$

The ANN model is trained on the training set. During the training process, the error of the updated model on the validation set is computed after each iteration to monitor the generalization performance. Once the validation error stagnates, this suggests that overfitting starts to develop, and the training process is terminated. The model is then tested on the test set, to make a final evaluation of the trained model.

Training the ANN is essentially an optimization process, which defines a cost function as

$$C(\boldsymbol{W}, \boldsymbol{B}) = \underbrace{\frac{1}{N^{\text{train}} N^O} \sum_{s=1}^{N^{\text{train}}} \left\| \hat{\boldsymbol{Y}}(\boldsymbol{X}^s; \boldsymbol{W}, \boldsymbol{B}) - \boldsymbol{Y}^s \right\|_2^2}_{\text{mean squared error}} + \underbrace{\lambda \left(\sum_{l=1}^{N^L} \frac{1}{N^{H_l} N^{H_{l-1}}} \left\| \boldsymbol{W}^{H_l} \right\|_2^2 + \frac{1}{N^O N^{H_{N^L}}} \left\| \boldsymbol{W}^O \right\|_2^2 \right)}_{\text{regularization}}, \tag{3.4}$$

where λ is a tunable positive parameter, N^{train} is the number of examples in the training set, $\boldsymbol{W} = \{\boldsymbol{W}^{H_1}, \ldots, \boldsymbol{W}^{H_{N^L}}, \boldsymbol{W}^O\}$, and $\boldsymbol{B} = \{\boldsymbol{B}^{H_1}, \ldots, \boldsymbol{B}^{H_{N^L}}, \boldsymbol{B}^O\}$. The first term on the right side is the mean squared error of the model on the training set, while the second is the regularization term. The goal of the optimization process is to minimize $C(\boldsymbol{W}, \boldsymbol{B})$, to achieve two things: 1. The mean squared error is minimized to generate an accurate model; 2. The weights are kept small to avoid overfitting. To better explain the second issue, we invoke a principle of parsimony known as *Occam's razor* [44], which states that among competing hypotheses, the one with the fewest assumptions should be selected. Small weights ensure that small changes in the input data bring small changes in the output, in accordance with *Occam's razor*.

In general, a key component in the optimization process is to compute partial derivatives of the cost function $C(\boldsymbol{W}, \boldsymbol{B})$ with respect to $\boldsymbol{W}$ and $\boldsymbol{B}$. By examining (3.4), it can be found that the only non-trivial part for computing partial derivatives lies in $\hat{\boldsymbol{Y}}(\boldsymbol{X}^s; \boldsymbol{W}, \boldsymbol{B})$. For convenience, we define $\boldsymbol{\nu} = \sigma(z)$ and $\delta = \frac{\partial \hat{\boldsymbol{Y}}}{\partial z}$. From (3.1), we have [40]

$$\begin{cases} \boldsymbol{\delta}^O &= \sigma'\left(\boldsymbol{z}^O\right) \circ \left(\hat{\boldsymbol{Y}} - \boldsymbol{Y}^s\right), \\ \boldsymbol{\delta}^{H_{N^L}} &= \sigma'\left(\boldsymbol{z}^{H_{N^L}}\right) \circ \left(\boldsymbol{W}^O\right)^T \boldsymbol{\delta}^O, \\ \boldsymbol{\delta}^{H_l} &= \sigma'\left(\boldsymbol{z}^{H_l}\right) \circ \left(\boldsymbol{W}^{H_{l+1}}\right)^T \boldsymbol{\delta}^{H_{l+1}}, \text{for } l = 1, \ldots, N^L - 1. \end{cases} \tag{3.5}$$

where the operator $\circ$ denotes the Hadamard product of two vectors. Then, if we let $C^s = \left\| \hat{\boldsymbol{Y}}(\boldsymbol{X}^s; \boldsymbol{W}, \boldsymbol{B}) - \boldsymbol{Y}^s \right\|_2^2$, the partial derivatives can be written as

$$\begin{cases} \dfrac{\partial C^s}{\partial B_m^{H_l}} = \delta_m^{H_l}, \text{for } l = 1, \ldots, N^L, \\ \dfrac{\partial C^s}{\partial B_m^O} = \delta_m^O, \end{cases} \tag{3.6}$$

$$\begin{cases} \dfrac{\partial C^s}{\partial W_{m,n}^{H_l}} = \delta_m^{H_l} \nu_n^{H_{l-1}}, \text{for } l = 1, \ldots, N^L, \\ \dfrac{\partial C^s}{\partial W_{m,n}^O} = \delta_m^O \nu_n^{H_{N^L}}. \end{cases} \tag{3.7}$$

From (3.6), we see that $\boldsymbol{\delta}^O$ is first obtained, and then $\boldsymbol{\delta}^{H_l}$ is computed in a backward direction. Therefore, this method of computing partial derivatives is also known as *back propagation*. For training, the weights $\boldsymbol{W}$ and biases $\boldsymbol{B}$ are first initialized, and then updated in each step as

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \Phi\left(\nabla^W C\right), \quad \boldsymbol{B} \leftarrow \boldsymbol{B} - \eta \Phi\left(\nabla^B C\right), \tag{3.8}$$

where η is the learning rate, $\nabla^W C$ and $\nabla^B C$ denote the vector of partial derivatives with respect to weights and biases, respectively, and Φ is related to the specific optimization algorithm. In this work, the so-called Adam algorithm[45] is employed, which has been widely used in training ANNs due to its fast convergence and robustness.

4. Data-driven models for shock capturing

In this section, data-driven shock capturing models suitable for one- and two-dimensional problems are proposed. An ANN is constructed on a number of given examples to map from solution to smoothness indicator, which in turn determines the artificial viscosity. In the following, we will first describe the model in the one-dimensional case, and then extend the idea to two spatial dimension.

4.1. One-dimensional case

4.1.1. Smoothness indicator

One key for training an ANN successfully is to use input and output features of compact data ranges, to emphasize approximation over extrapolation [30]. For the input, one can use normalization to ensure compactness, while for the output, a suitable indicator is necessary. In the following, we will employ the smoothness indicator given in [29], which proposed a polynomial analogy to Fourier expansions, i.e. the modal decay could be represented as

$$| \hat{q}_m | \simeq c m^{-\tau}, \tag{4.1}$$

where τ denotes the modal decay rate, and serves as the smoothness indicator. If the analogy to the Fourier case holds up,

$$\tau = \begin{cases} 1 & \text{if } q \text{ is discontinuous,} \\ 2 & \text{if } q \in C^0 \backslash C^1, \\ 3 & \text{if } q \in C^1 \backslash C^2, \\ \cdots & . \end{cases} \tag{4.2}$$

Based on this smoothness indicator, the artificial viscosity is computed as

$$\mu = \mu^{\max} \begin{cases} 1 & \text{if } \tau < 1, \\ 1 - (\tau - 1)/2 & \text{if } 1 \le \tau \le 3, \\ 0 & \text{if } \tau > 3. \end{cases} \tag{4.3}$$

Here, $\mu^{\max}$ denotes the full viscosity, given as

$$\mu^{\max} = c^{\max} (h/P) \max_{\boldsymbol{x} \in G_K} | \boldsymbol{f}'(q_h(\boldsymbol{x}, t)) |, \tag{4.4}$$

where $c_{\max}$ is an empirical parameter, G_K denotes the set of nodal points within element K, and $\boldsymbol{f}'(q_h(\boldsymbol{x}, t))$ is the local wave speed. In this work, $c_{\max}$ is fixed to 1.

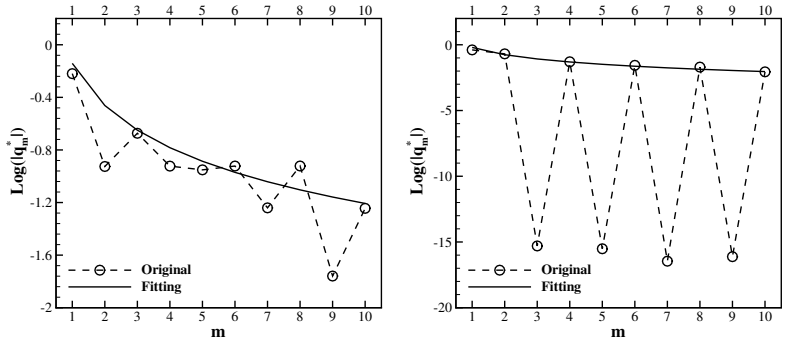
In [29], τ is computed from the modal coefficients, which frequently exhibit oscillatory behavior as illustrated in Fig. 3. This non-monotonicity interferes with the determination of the decay rate when using a least-squared method. A couple of fixes were designed in [29], based on estimates of the smoothness in an averaged sense, and we will refer to it as averaged modal decay (MDA) in the following. In this work, a direct map from q to τ is built with an ANN. From (4.3), we see that the effective value of τ is restricted to $[1, 3]$, since any value outside this range can be reset to either the left or the right limit and generate the same viscosity. Consequently, the output feature τ is guaranteed to be compact.

Remark 1. The artificial viscosity computed by (4.3) is piecewise constant, and needs to be smoothed to enhance robustness and accuracy. A $C0$ smoothing technique is employed and described below.

Step 1. Choose a node set corresponding to P2 for each element, and average along each edge node for the cells sharing the same node.

Step 2. Within each element, construct a Lagrange polynomial of second order, based on the averaged nodal values.

Step 3. Artificial viscosity with $C0$ continuity at any position can be computed using this element-wise polynomial.



(a) $H(x)$ (b) $xH(x)$

(c) $x^2H(x)$ (d) $\sin(x)$

Fig. 3. Modal coefficients and fitting curves for a standard $P10$ element ($x \in [-1, 1]$) with various functions. $H(x)$ denotes the Heaviside jump function. For original coefficients, $q_m^* = \hat{q}_m$, while for the fitting curve, q_m^* denotes the modified modal coefficients computed by the approach given in [29].

4.1.2. Sampling and training

A desirable property of the smoothness indicator is that one can determine the decay rate of known functions explicitly according to (4.2). Table 1 presents the functions used to generate the training dataset. For functions that are expected to be smooth and should need no artificial viscosity, i.e., F_2, F_3, F_4, F_5, the decay rates are set to 4. F_3 is included to ignore small wiggles. Note that the functions used for sampling are not limited to those listed in Table 1, and other choices are also possible.

Table 1. Sampling functions with selected parameters

Function ($x \in [-1, 1]$)	Selected parameters	τ
$F_0(x; c^L, c^R, x_0) = c^L \lfloor x < x_0 \rfloor^* + c^R \lfloor x \geq x_0 \rfloor$	$\{c^L, c^R\}^{[m]} \sim \mathbb{U}([-10^3, 10^3])^{**}, m = 1, \ldots, 1000,$ $x_0^{[n]} = \frac{r_n + r_{n+1}}{2}^{***}, n = 0, \ldots, P-1.$	1
$F_1(x; c^L, c^R, x_0) = (x - x_0)\left(c^L \lfloor x < x_0 \rfloor + c^R \lfloor x \geq x_0 \rfloor\right)$	$\{c^L, c^R\}^{[m]} \sim \mathbb{U}([-10, 10]) \quad, m = 1, \ldots, 1000,$ $x_0^{[n]} = -1 + 2\frac{n+1}{P+1}, n = 0, \ldots, P-1.$	2
$F_2(x; c) = c$	$c^{[m]} = 1, \quad m = 1.$	4
$F_3(x; c) = c + \epsilon(x), \quad \epsilon(x) \sim \mathbb{U}([-10^{-4}, 10^{-4}])$	$c^{[m]} \sim \mathbb{U}([-1, 1]), \quad m = 1, \ldots, 1000.$	4
$F_4(x; c, x_0) = c(x - x_0)$	$c^{[m]} \sim \mathbb{U}([-30, 30]),$ $x_0^{[m]} \sim \mathbb{U}([-1, 1]), \quad m = 1, \ldots, 1000.$	4
$F_5(x; c, x_0) = \sin\left(2\pi\frac{x-x_0}{c}\right)\cos\left(3\pi\frac{x-x_0}{c}\right)\sin\left(4\pi\frac{x-x_0}{c}\right)$	$c^{[m]} = \frac{2^{1-m}}{10}, \quad m = 1, \ldots, 6,$ $x_0^{[n]} = c^{[m]}\left(n + \frac{1}{2}\right), \quad n = 1, \ldots, 10 \times 2^m - 2.$	4

* $\lfloor \bullet \rfloor$ denotes the Iverson bracket.
** This indicates that c^L, c^R are random variables uniformly distributed on $[-10^3, 10^3]$.
*** r_n denotes the nth Legendre-Gauss-Lobatto point within an element.

Let us describe how to prepare the samples according to Table 1. For the sth sample $\boldsymbol{\Xi}_s$ (see (3.2)), $\boldsymbol{Y}^s \in \mathbb{R}$ is the decay rate τ. For $\boldsymbol{X}^s$, the following steps are conducted

Step 1. Nodal values $\tilde{\boldsymbol{q}}$ on the standard 1D element are computed using the functions equipped with the selected parameters given in Table 1.

Step 2. The nodal values are normalized as $\tilde{\boldsymbol{q}} \leftarrow \tilde{\boldsymbol{q}}/q^{\max}$, where $q^{\max} = \max_{m=0,\ldots,P} | \tilde{q}_m |$.

Step 3. The modal coefficients are computed with the normalized nodal values on this element using (2.7).

Step 4. Solution values $\left(q_m^h, m = 0, \ldots, 10\right)$ at 11 uniformly distributed points (including both end points) within the same element are obtained using the modal expression (2.5). The input vector $\boldsymbol{X}^s$ is given as

$$\boldsymbol{X}^s = \begin{bmatrix} q_0^h \\ \vdots \\ q_{10}^h \end{bmatrix} \in \mathbb{R}^{11}. \tag{4.5}$$

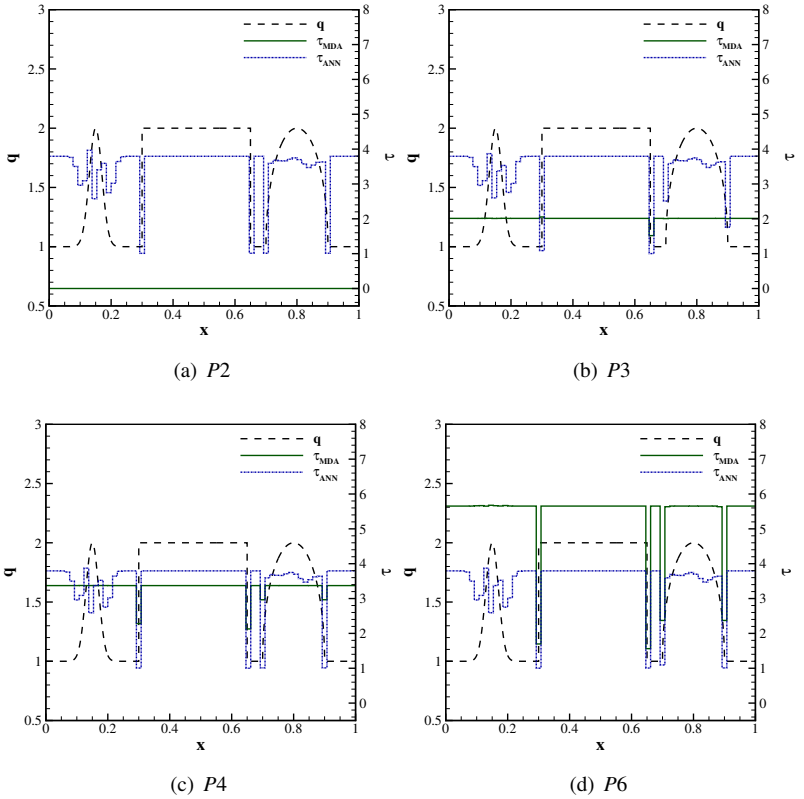
Following (3.3), the samples are divided into three groups, i.e. $\boldsymbol{\Xi}^{\text{train}}$, $\boldsymbol{\Xi}^{\text{test}}$ and $\boldsymbol{\Xi}^{\text{validation}}$, such that the training set takes 80% of all the samples, while the test and validation sets bisect the rest. In this work, three hidden layers are employed with $N^{H_1} = 32$, $N^{H_2} = 16$ and $N^{H_3} = 8$. The learning rate η in (3.8) is fixed to 10^{-4}. The weights $\boldsymbol{W}$ and biases B are initialized with random numbers uniformly distributed in $[0, 1]$, and the model with the smallest error among 20 trainings are selected to overcome the randomness caused by initialization.

4.1.3. Testing the ANN

In this section, we test the trained ANN by examining the prediction of the smoothness indicator for the following function

$$q(x) = \begin{cases} 1 + e^{-300(2x-0.3)^2} & \text{if } |x - 0.15| \leq 0.1, \\ 2 & \text{if } |x - 0.475| \leq 0.175, \\ 1 + \sqrt{|1 - (10x - 8)^2|} & \text{if } |x - 0.8| \leq 0.1, \\ 1 & \text{otherwise.} \end{cases} \tag{4.6}$$

The domain is chosen to be $[0, 1]$ with 65 uniform elements. The MDA model is also included for comparison. As is seen in Fig. 4, the ANN-based model predicts the decay rate well for all the cases, i.e., $\tau_{\text{ANN}} \approx 1$ around discontinuities and $\tau_{\text{ANN}} \approx 4$ in smooth regions. For the sharp pulse around $x = 0.15$, ANN tends to slightly underestimate the smoothness due to the coarse resolution. This disappears on finer grids as discussed further in Section 5.1. On the contrary, MDA is observed to be less reliable. In general, the decay rate obtained by MDA in constant regions is close to P due to the technique of introducing baseline modal decay, and excessive dissipation occurs in such regions for lower orders (lower than $P4$). Furthermore, the oscillatory behavior of the modal coefficients, illustrated in Fig. 3, requires another technique called skyline pessimization, which enforces the coefficients to decay monotonically. However, this alters the original function, and introduces some uncertainty in the final decay rate especially for low orders as shown in Fig. 4.



(a) $P2$ (b) $P3$

(c) $P4$ (d) $P6$

Fig. 4. Comparison on predicted decay rates of a complex function.

Computational costs of the two artificial viscosity models are compared in Fig. 5. The cost of MDA increases significantly with polynomial orders, while ANN is hardly affected by polynomial orders. From $P3$ and up, the cost of the ANN is comparable or less. Note that MDA is unable to deliver reasonable results for polynomial orders of lower than 4 (see Fig. 4).

4.2. Two-dimensional case

For two-dimensional cases, we apply the one-dimensional model along each edge of the triangles. In a triangle, a discontinuity is expected to intersect at least one edge. Hence, one can estimate the smoothness of the triangle by examining the smoothness of the one-dimensional solutions along the three edges. The detailed steps extending the one-dimensional model to two-dimensional cases are presented as follows

Step 1. Extract the nodal values along each edge of the triangle. Three one-dimensional problems are formed in a straightforward manner considering the nodes of the triangles.

Step 2. Estimate a decay rate along each edge with the one-dimensional ANN model, and choose the largest one as the decay rate on the triangle.

Step 3. Compute an element-wise artificial viscosity according to (4.3).

Step 4. Smooth the element-wise viscosity with the same technique as described in 1D.

Remark 2. The proposed method is expected to extend to arbitrary two- and three-dimensional elements in a similar manner.

5. Results

In the following, we consider a number of typical test cases to investigate the performance of the proposed shock capturing model. The effect of the model on smooth regions is first examined with convergence tests on one- and two-dimensional smooth problems. Then several cases with discontinuities including the one-dimensional Burgers' problem, one-dimensional shock tube problems, the Shu-Osher problem, the two-dimensional Riemann problems, the double Mach problem and the forward step problem are performed to show the robustness of the model for shock capturing.

5.1. Convergence tests on smooth problems

5.1.1. One-dimensional linear transport

For the one-dimensional case, a linear transport is considered to investigate the convergence property of the proposed model for smooth flows. The flux in (2.1) is $f = q$ with an exact solution being $q(x,t) = \sin(\pi(x-t))$, defined on a domain of $[0, 2\pi]$. The numerical error is computed after two periods, i.e., $t = 4$. Results are presented in Table 2, where linear schemes are also included for comparison. As it can be seen, the proposed shock capturing model recovers accuracy of the corresponding linear scheme on a sufficiently fine grid, validating the discussions in Fig. 4.

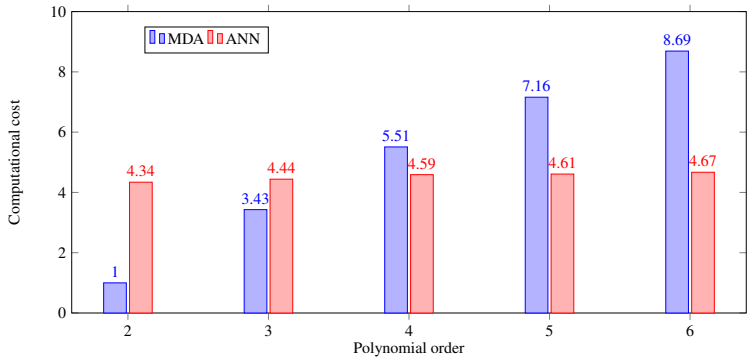


Fig. 5. Comparison on computational costs of artificial viscosity models. Costs are relative to that of MDA at $P2$.

Table 2. L^2 errors and orders of accuracy for the one-dimensional scalar case.

	N^g	Linear				ANN			
		L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
$P2$	10	1.71E-03		3.85E-03		6.62E-02		1.16E-01	
	20	2.17E-04	2.98	5.08E-04	2.92	2.17E-04	8.26	5.08E-04	7.83
	40	2.72E-05	3.00	6.43E-05	2.98	2.72E-05	3.00	6.43E-05	2.98
	80	3.40E-06	3.00	8.07E-06	3.00	3.40E-06	3.00	8.07E-06	3.00
	160	4.25E-07	3.00	1.01E-06	3.00	4.25E-07	3.00	1.01E-06	3.00
$P3$	10	6.93E-05		1.85E-04		5.63E-02		1.14E-01	
	20	4.33E-06	4.00	1.15E-05	4.01	4.33E-06	13.7	1.15E-05	13.3
	40	2.71E-07	4.00	7.23E-07	3.99	2.71E-07	4.00	7.23E-07	3.99
	80	1.69E-08	4.00	4.53E-08	4.00	1.69E-08	4.00	4.53E-08	4.00
	160	1.06E-09	4.00	2.83E-09	4.00	1.06E-09	4.00	2.83E-09	4.00
$P4$	10	2.18E-06		6.23E-06		4.22E-02		9.34E-02	
	20	6.83E-08	4.99	1.98E-07	4.98	6.83E-08	19.2	1.98E-07	18.9
	40	2.16E-09	4.98	6.30E-09	4.97	2.16E-09	4.98	6.30E-09	4.97
	80	6.60E-11	5.03	1.90E-10	5.05	6.60E-11	5.03	1.90E-10	5.05

5.1.2. Two-dimensional isentropic vortex

In this section, the convergence test is continued with the two-dimensional isentropic vortex, governed by the two-dimensional Euler system. The corresponding variables and fluxes in (2.1) are

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \qquad \boldsymbol{f} = \boldsymbol{f}^x \vec{\boldsymbol{i}} + \boldsymbol{f}^y \vec{\boldsymbol{j}}, \tag{5.1}$$

and

$$\boldsymbol{f}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \qquad \boldsymbol{f}^y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}, \tag{5.2}$$

where ρ is the density, E is the total energy, u and v represent the velocities in the x and y directions, respectively. p denotes the pressure with $p = (\gamma - 1)\left(E - \frac{1}{2}\rho\left(u^2 + v^2\right)\right)$ and $\gamma = 1.4$. The initial condition for this case is

$$\left.\begin{array}{ll} \rho = T^{\frac{1}{\gamma-1}}, & p = \rho^\gamma, \\ u = 1 - \frac{5}{2\pi}\exp\left(\frac{1-r^2}{2}\right)(y - y_0), & v = 1 + \frac{5}{2\pi}\exp\left(\frac{1-r^2}{2}\right)(x - x_0), \\ T = 1 - \frac{25(\gamma-1)}{8\gamma\pi^2}\exp(1 - r^2), & r = \sqrt{(x - x_0)^2 + (y - y_0)^2}, \end{array}\right\} \tag{5.3}$$

with $(x_0, y_0) = (-5, -5)$ on a domain of $[-10, 10] \times [-10, 10]$. The exact solution is the initial vortex convecting at a velocity vector of $(1, 1)$, and the error is evaluated at $t = 10$. The grids are obtained by dividing each quadrilateral element of uniform structured grids into two triangles. The convergence results are presented in Table 3. Similar to the one-dimensional case, the data-driven model is triggered on the coarsest grids only, and recovers the accuracy of the linear scheme as the refinement increases.

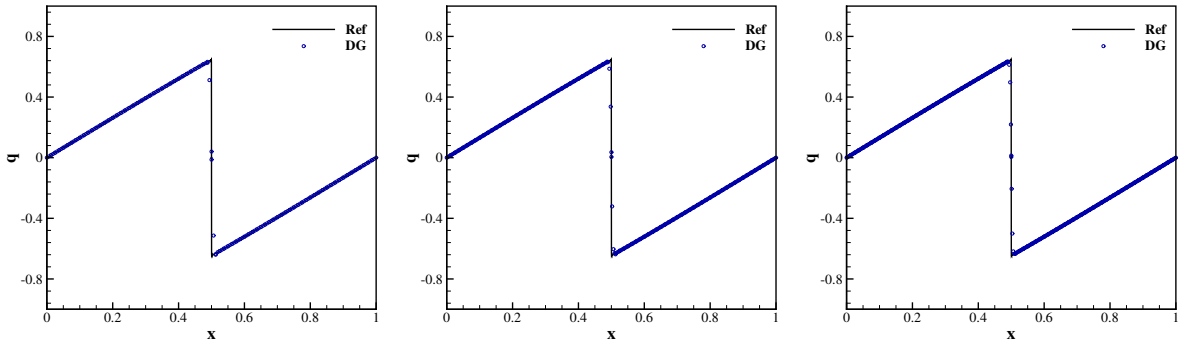
5.2. One-dimensional Burgers' problem

Let us now apply the proposed model to discontinuous flows. The first one is the one-dimensional Burgers' problem, for which the corresponding flux is $f = q^2$. The computational domain is $[0, 1]$, and the initial condition is

Table 3. L^2 errors and orders of accuracy for the two-dimensional isentropic vortex.

	N^g	Linear				ANN			
		L^2 error	Order	L^∞ error	Order	L^2 error	Order	L^∞ error	Order
$P2$	40	8.64E-04		1.10E-02		1.03E-02		2.12E-01	
	80	1.54E-04	2.49	3.30E-03	1.74	7.07E-04	3.87	2.73E-02	2.96
	160	3.46E-05	2.16	1.29E-03	1.35	3.46E-05	4.35	1.29E-03	4.40
	320	8.09E-06	2.10	4.27E-04	1.60	8.09E-06	2.10	4.27E-04	1.60
	640	1.72E-06	2.23	1.11E-04	1.94	1.72E-06	2.23	1.11E-04	1.94
$P3$	40	3.18E-04		3.06E-03		1.03E-02		2.16E-01	
	80	1.18E-05	4.75	1.80E-04	4.09	8.04E-04	3.68	3.06E-02	2.82
	160	5.96E-07	4.31	1.22E-05	3.88	5.96E-07	10.4	1.22E-05	11.3
	320	3.84E-08	3.95	1.20E-06	3.34	3.84E-08	3.95	1.20E-06	3.34
	640	2.58E-09	3.90	1.23E-07	3.29	2.58E-09	3.90	1.23E-07	3.29
$P4$	40	2.40E-05		4.44E-04		9.82E-03		2.09E-01	
	80	1.17E-06	4.36	5.13E-05	3.12	7.67E-04	3.68	2.93E-02	2.83
	160	8.62E-08	3.76	4.46E-06	3.52	8.62E-08	13.1	4.46E-06	12.68
	320	5.17E-09	4.06	3.74E-07	3.58	5.17E-09	4.06	3.74E-07	3.58

$q(x,0) = \sin(2\pi x)$, from which a discontinuity gradually develops in the middle of the domain. 80 uniform elements are used and the computation is terminated at $t = 0.3$. Solutions of DG with the proposed shock capturing model are plotted in Fig. 6, from which it is clear that the discontinuity is captured well with no spurious oscillations. Note that each symbol corresponds to a value on a node of an element, i.e., the full solutions are plotted in this work unless specified otherwise. Furthermore, we present the artificial viscosity generated by the model in Fig. 7, which clearly shows the development of the discontinuity. The peak value of the viscosity decreases with increasing orders as expected. The slight asymmetry of the artificial viscosity is attributed to the randomness of the samples in Table 1.



(a) $P2$ (b) $P4$ (c) $P6$

Fig. 6. Solution for the one-dimensional Burgers' problem with 80 elements at $t = 0.3$.

5.3. One-dimensional Euler system

In this section, several one-dimensional Euler problems are considered, and the corresponding governing equation can be obtained in a straight-forward manner from (5.1).

5.3.1. Shock tube problems

Shock tube problems are widely used for testing shock capturing methods. The computation domain is chosen to be $[0, 1]$ with 100 uniform elements, and two initial conditions are considered, i.e., the Sod problem, (5.4), and the Lax problem, (5.5).

$$(\rho, u, p) = \begin{cases} (1, 0, 1) & x < 0.5, \\ (0.125, 0, 0.1) & x > 0.5. \end{cases} \tag{5.4}$$

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528) & x < 0.5, \\ (0.5, 0, 0.571) & x > 0.5. \end{cases} \tag{5.5}$$

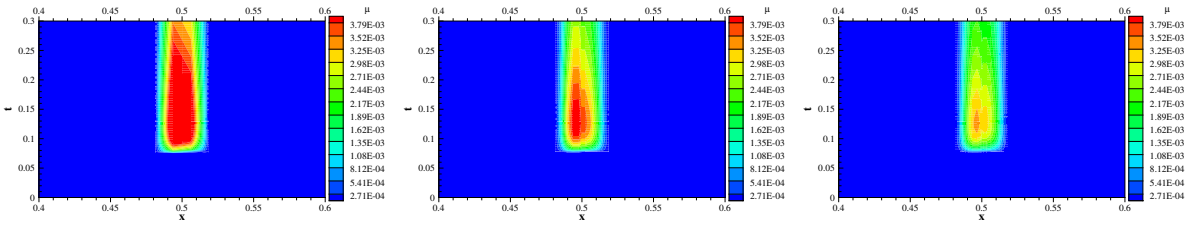
Density profiles are examined at $t = 0.2$ and $t = 0.13$ for the Sod (Fig. 8) and Lax (Fig. 9) problems, respectively. As can be seen, the proposed model suppresses spurious oscillations well.

5.3.2. Shu-Osher problem

The Shu-Osher problem is an one-dimensional model for shock/turbulence interactions and requires the numerical method to simultaneously deal with shocks and small fluctuations. The computational domain is $[-5, 5]$, discretized with uniform elements. The initial condition is given by

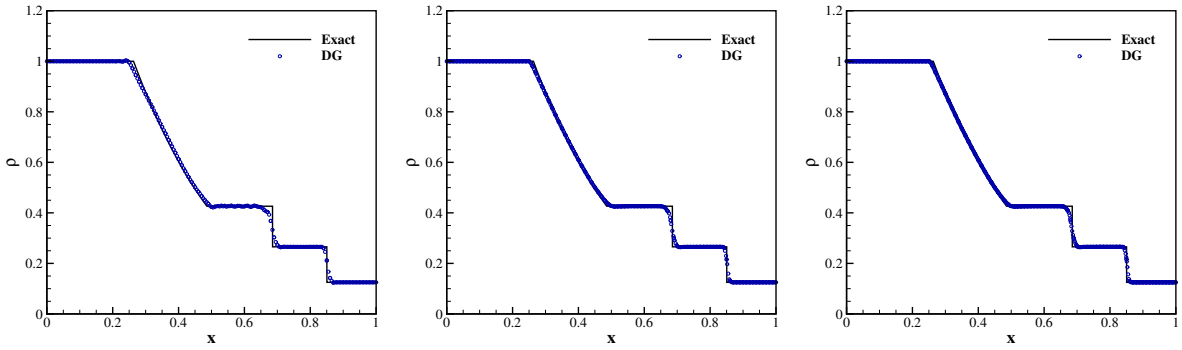
$$(\rho, u, p) = \begin{cases} (3.857143, 2.629369, 10.333333) & x < -4, \\ (1.0 + 0.2\sin(5x), 0, 1) & x > -4. \end{cases} \tag{5.6}$$

Density profiles obtained with DG of typical orders are presented in Figs. 10 and 11, where shocks are captured sharply with no spurious oscillations. On the finer grid, the results almost coincide with the reference solution. In Fig. 12, we show the temporal development of artificial viscosity for the coarser grid. Note that the logarithm of the



(a) $P2$ (b) $P4$ (c) $P6$

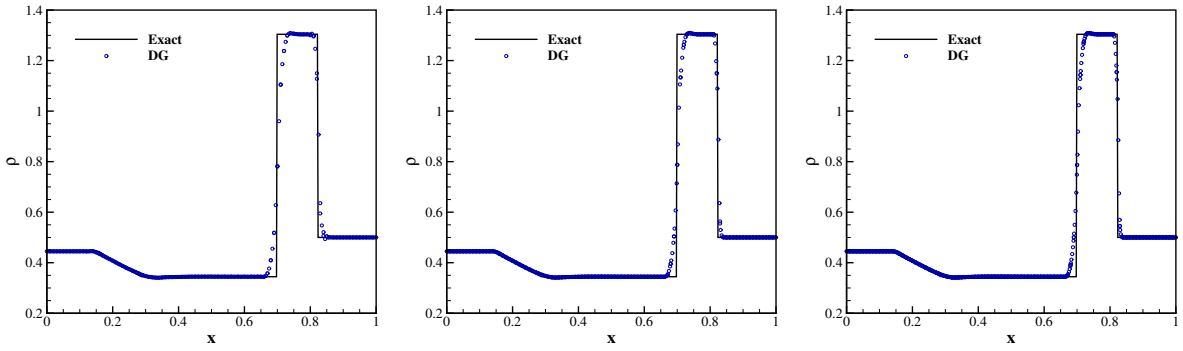
Fig. 7. Artificial viscosity for the one-dimensional Burgers' problem with 80 elements.



(a) $P2$ (b) $P4$ (c) $P6$

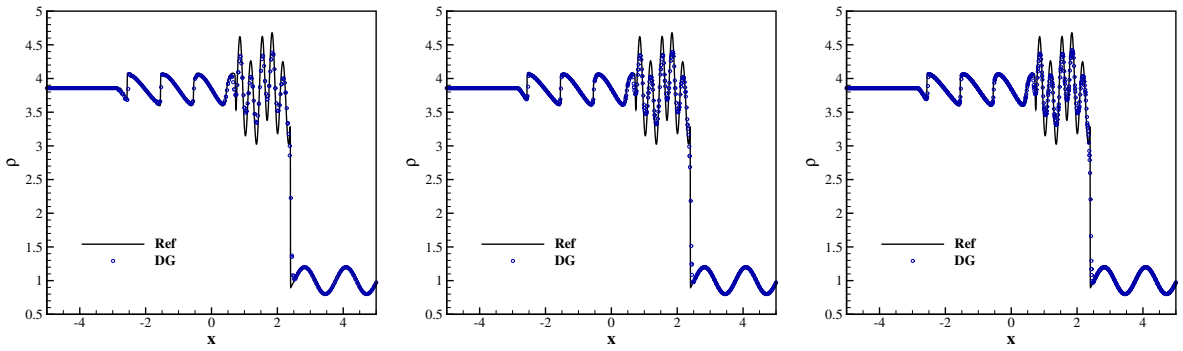
Fig. 8. Density profiles for the Sod problem with 100 elements at $t = 0.2$.

viscosity is plotted to obtain a better visualization. As is clear, the distribution of viscosity agrees well with shocks and extrema. Note that the dissipation for extrema in our case can be eliminated with grid refinement as discussed in Fig. 4 shown in Tables 2 and 3.



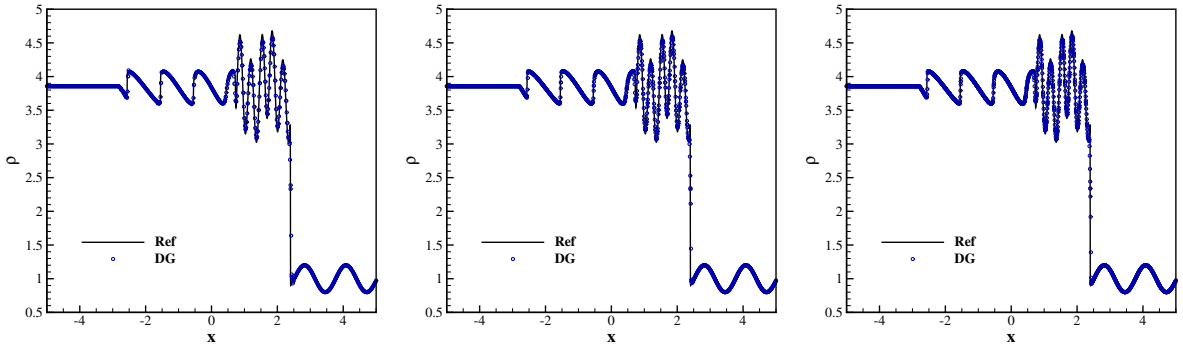
(a) $P2$ (b) $P4$ (c) $P6$

Fig. 9. Density profiles for the Lax problem with 100 elements at $t = 0.13$.



(a) $P2$ (b) $P4$ (c) $P6$

Fig. 10. Density profiles for the Shu-Osher problem with 160 elements at $t = 1.8$.



(a) $P2$ (b) $P4$ (c) $P6$

Fig. 11. Density profiles for the Shu-Osher problem with 300 elements at $t = 1.8$.

5.4. Two-dimensional Euler system

In this section, the proposed shock capturing model is tested against several two-dimensional problems governed by the two-dimensional Euler system.

5.4.1. 2D Riemann problem

Among various 2D Riemann problems [46], Case 4 and Case 12 are studied here on a domain of $[0,1]\times[0,1]$ with initial conditions given by (5.7) (Case 4) and (5.8) (Case12), respectively. The grids are obtained by dividing each quadrilateral element of a uniform structured grid into two triangles. For both cases, the results are shown at $t = 0.25$ in Figs. 13 and 14. As can be seen, the proposed model is able to suppress spurious oscillations and preserves contact lines well.

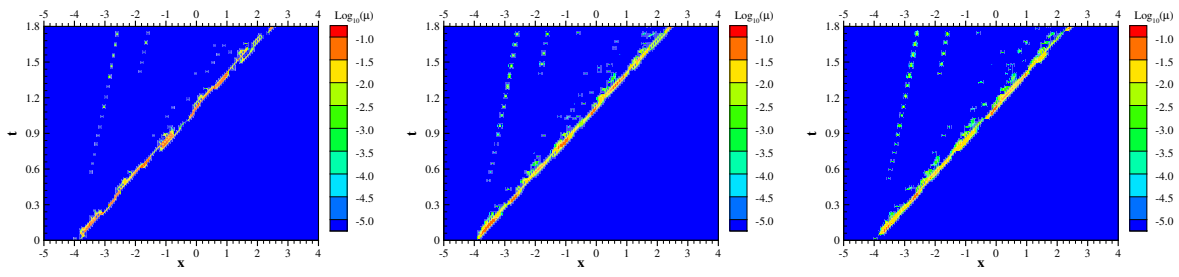
$$(\rho,u,v,p) = \begin{cases} (1.1,0,0,1.1) & 0.5 < x < 1, 0.5 < y < 1, \\ (0.5065,0.8939,0.0,0.35) & 0 < x < 0.5, 0.5 < y < 1, \\ (1.1,0.8939,0.8939,1.1) & 0 < x < 0.5, 0 < y < 0.5, \\ (0.5065,0,0.8939,0.35) & 0.5 < x < 1, 0 < y < 0.5. \end{cases} \tag{5.7}$$

$$(\rho,u,v,p) = \begin{cases} (0.5313,0,0,0.4) & 0.5 < x < 1, 0.5 < y < 1, \\ (1.0,0.7276,0.0,1.0) & 0 < x < 0.5, 0.5 < y < 1, \\ (0.8,0,0,1) & 0 < x < 0.5, 0 < y < 0.5, \\ (1.0,0,0.7276,1.0) & 0.5 < x < 1, 0 < y < 0.5. \end{cases} \tag{5.8}$$

5.4.2. Double Mach problem

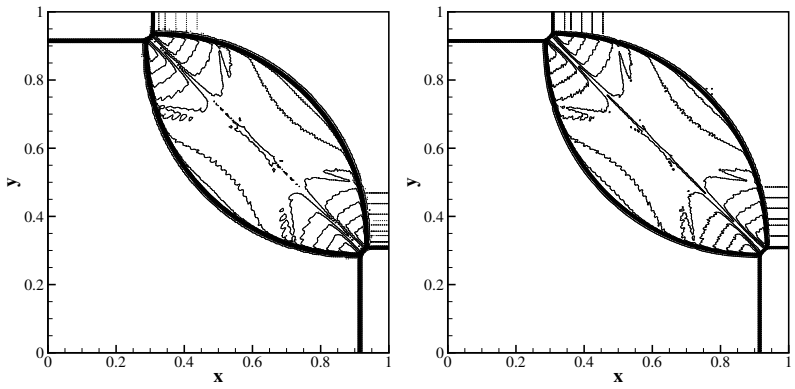
The double Mach problem is widely used for investigating the performance of numerical schemes for strong shocks [47]. The computation is conducted in a rectangular domain of $[0,4]\times[0,1]$, where an inviscid wall is placed at the bottom of the domain for $x \geq 1/6$. A Mach 10 shock, which makes a 60° angle with the horizontal line, moves to the right, starting at $x = 1/6, y = 0$. Therefore, the conditions for the left boundary and the part from $x = 0$ to $x = 1/6$ on the bottom boundary are prescribed the post-shock conditions. At the top boundary, flow variables are set to describe the motion of the Mach 10 shock. Outflow conditions are set at the right boundary. The grids are obtained in the same manner as the two-dimensional Riemann case, and the computation is stopped at $t = 0.2$.

In Fig. 15, we present density contours obtained with $P2$ and $P4$ on a grid of $h = 1/204$.Strong shocks are captured well with little spurious oscillations. To better examine the small wiggles in smooth regions, we plot the cell-averaged results in Fig. 16. As can be seen, the results are much smoother, indicating that most wiggles reside at the element level. In Fig. 17, we further compare the results around the double Mach region, which clearly shows that more noticeable roll-up of contact lines is observed with higher orders on the same grid. Fig. 18 shows that the proposed model generates viscosity in good agreement with the distribution of the flow smoothness.



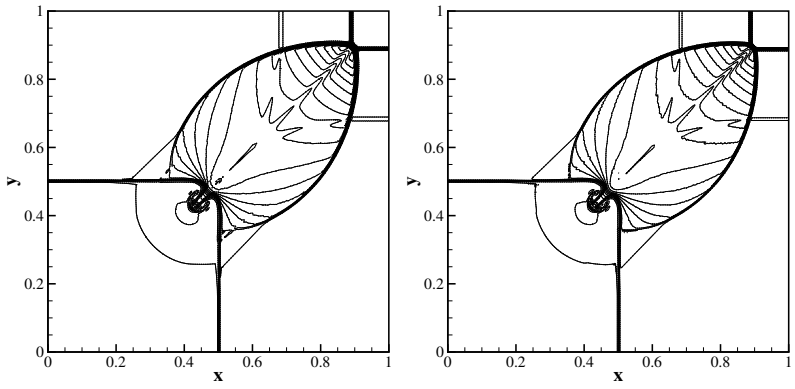
(a) $P2$ (b) $P4$ (c) $P6$

Fig. 12. Artificial viscosity for the Shu-Osher problem with 160 elements at $t = 1.8$.



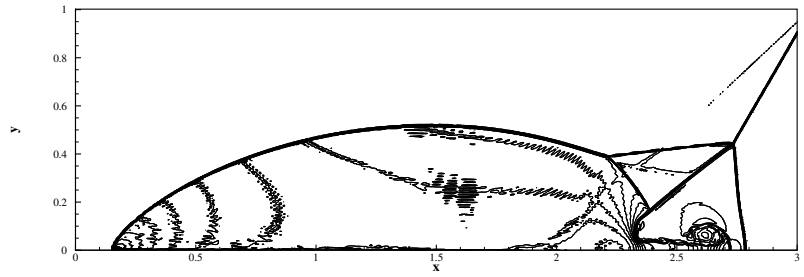
(a) $P2$ (b) $P4$

Fig. 13. Density contours for the two-dimensional Riemann problem (Case 4) at $t = 0.25$ ($h = 1/160$). Thirty equally spaced contour lines from 0.255 to 1.9.

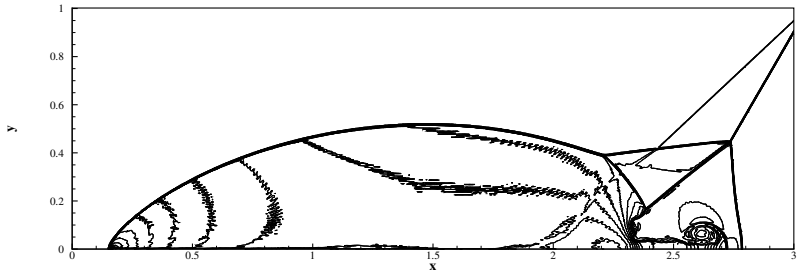


(a) $P2$ (b) $P4$

Fig. 14. Density contours for the two-dimensional Riemann problem (Case 12) at $t = 0.25$ ($h = 1/160$). Thirty equally spaced contour lines from 0.515 to 1.665.

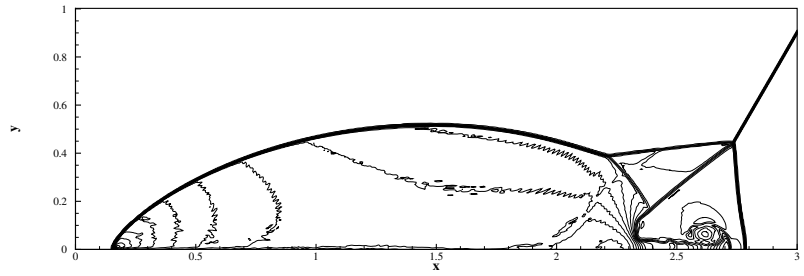


(a) $P2$

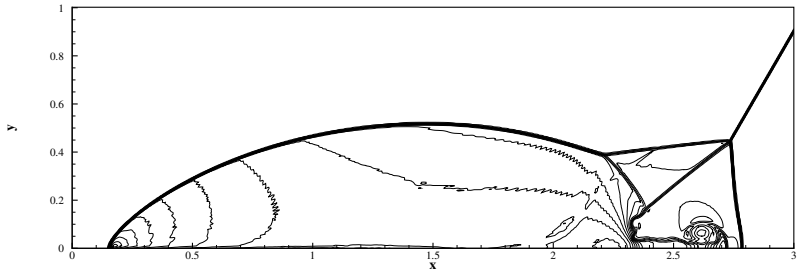


(b) $P4$

Fig. 15. Density contours for the double Mach problem at $t = 0.2$ ($h = 1/204$). Thirty equally spaced contour lines from 1.85 to 22.69.

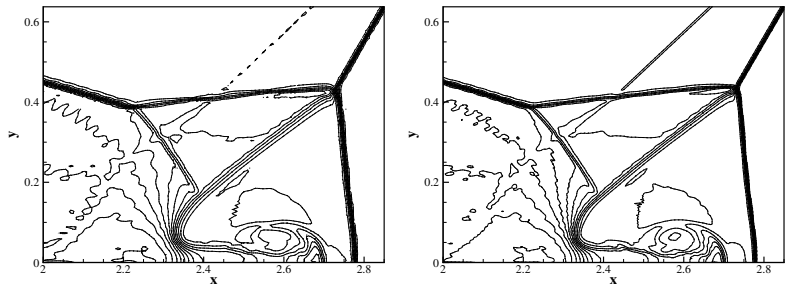


(a) $P2$

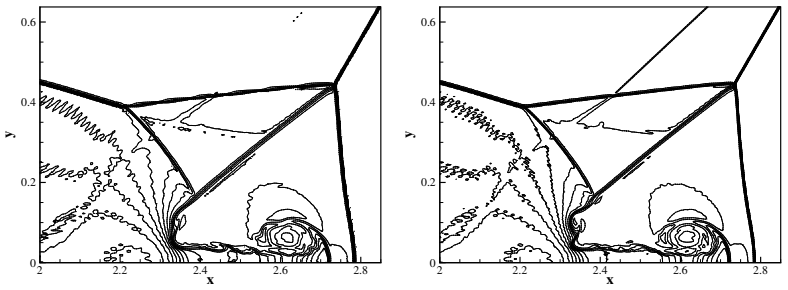


(b) $P4$

Fig. 16. Cell-averaged density contours for the double Mach problem at $t = 0.2$ ($h = 1/204$). Thirty equally spaced contour lines from 1.85 to 22.69.

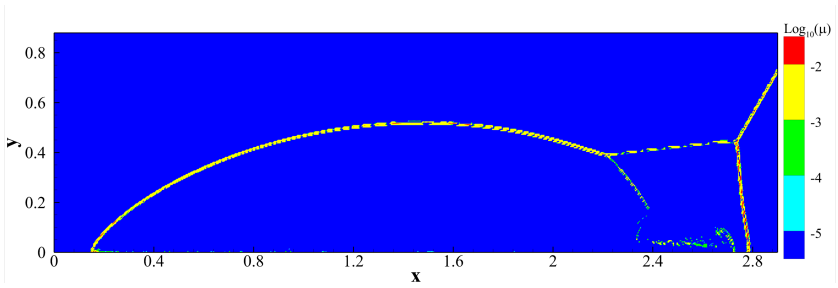


(a) $P2$, $h = 1/102$ (b) $P4$, $h = 1/102$

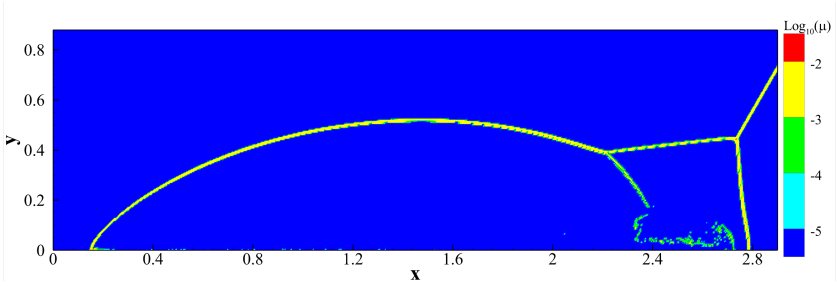


(c) $P2$, $h = 1/204$ (d) $P4$, $h = 1/204$

Fig. 17. Density contours for the double Mach problem at $t = 0.2$. Thirty equally spaced contour lines from 1.85 to 22.69.



(a) $P2$



(b) $P4$

Fig. 18. Artificial viscosity contours for the double Mach problem at $t = 0.2$ ($h = 1/204$).

5.4.3. Forward step problem

This problem describes a Mach 3 flow past a step in a wind tunnel, the geometry of which is shown in Fig. 19, along with a fully unstructured grid. The grids used for the results presented in this section are similar to that shown in Fig. 19 with different element sizes. Following [27], density contours are shown at $t = 3$ in Fig. 20. Note that a challenging aspect with this case is the singular point of the step corner, for which no special treatment is employed in our work. As can be seen, the proposed model is able to stabilize shocks on fully unstructured grids. With the polynomial order increasing, delicate roll-up of contact lines originating from the triple-wave point becomes more noticeable on the same grid. The entropy layer along the horizontal wall of the step is triggered by the singularity of the corner, and is reduced with order increasing. Furthermore, the artificial viscosity is observed to agree well with the flow structures in Fig. 21.

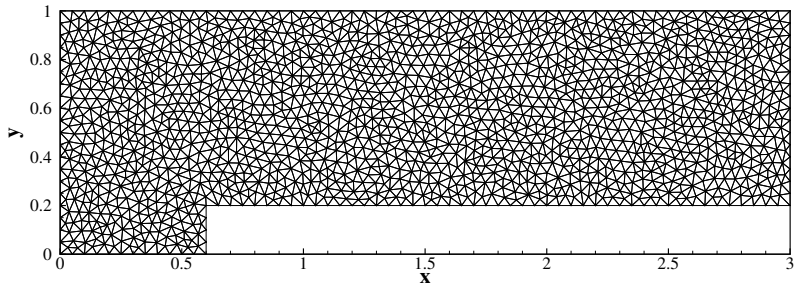
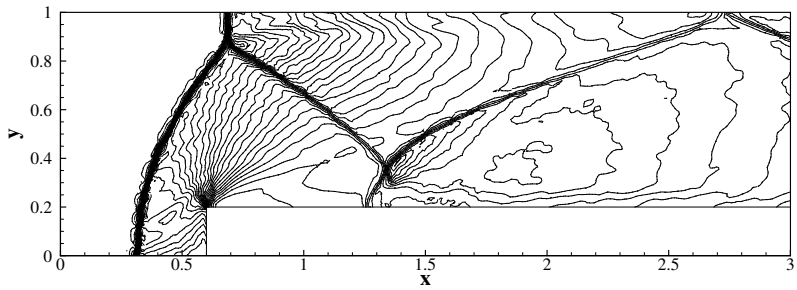
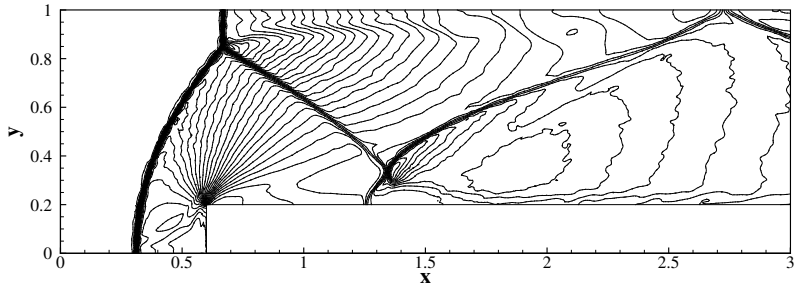


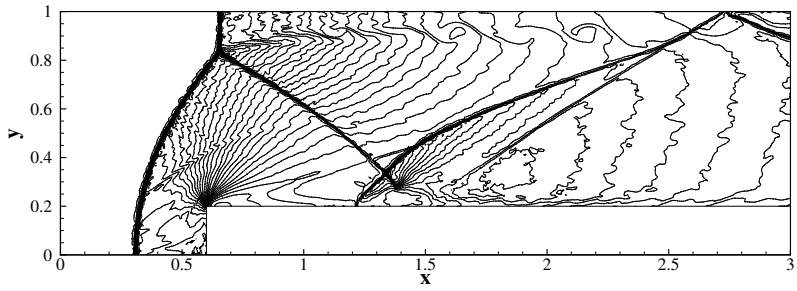
Fig. 19. A sample grid for the forward step problem($h = 1/20$).



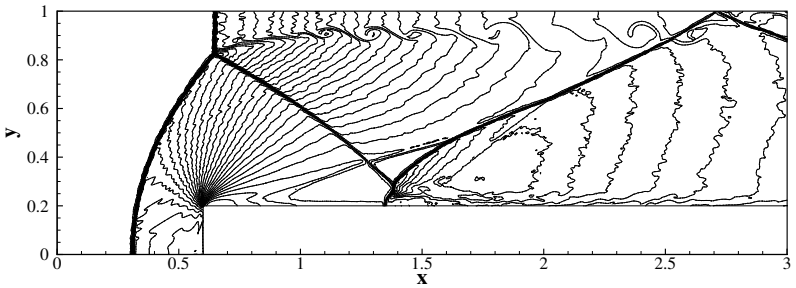
(a) $P2$, $h = 1/40$



(b) $P4$, $h = 1/40$

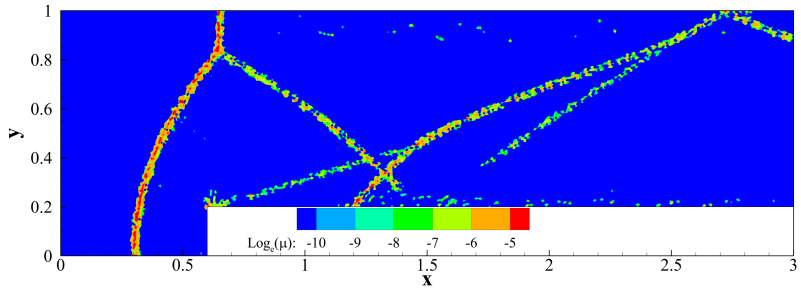


(c) $P2$, $h = 1/80$

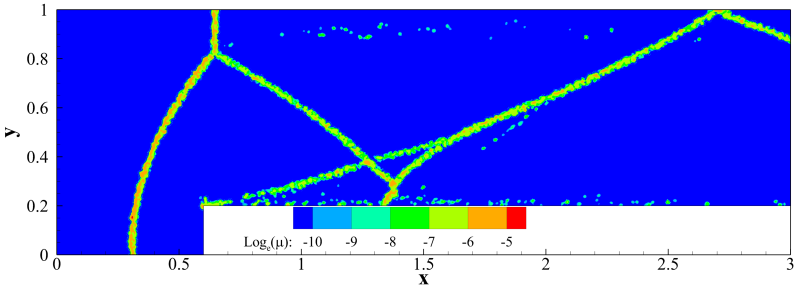


(d) $P4$, $h = 1/80$

Fig. 20. Density contours for the forward step problem at $t = 3$. Thirty equally spaced contour lines from 0.42 to 6.466.



(a) $P2$



(b) $P4$

Fig. 21. Artificial viscosity contours for the forward step problem at $t = 3$ ($h = 1/80$).

6. Conclusion

A data-driven artificial viscosity model for shock capturing with the discontinuous Galerkin method is proposed. A multi-layer feedforward network, mapping the element-wise solution to a smoothness indicator is trained on a data set of typical functions with smoothness indicators explicitly identified. Then, the artificial viscosity is computed based on the smoothness indicator delivered by the network. The normalization of the input data and the selection of the smoothness indicator for the output are able to ensure compactness, which is critical to the success of the training of the network. The final network is observed to recover the smoothness of canonical functions more reliably than the original MDA model.

A number of canonical cases are studied to investigate the performance of the proposed model for both smooth and discontinuous flows. Convergence tests with one-dimensional linear transport and two-dimensional isentropic vortex convection confirm that the model impacts the accuracy on the coarsest grids. However, as the grids are refined, the model is observed to recover the accuracy of the corresponding linear schemes. For non-smooth flows, the proposed model captures shocks with no spurious oscillations. The resolution and accuracy are seen to increase with the polynomial order on the same grid. The distribution of the artificial viscosity agrees with the non-smoothness of the flows. For discontinuous flows, $P2$ tends to be more oscillatory than its higher-order counterparts, due to insufficient information of smoothness contained in lower order polynomials. However, good results are obtained overall. The applicability of the data-driven model to viscous flows requires further investigation, which constitutes of our on-going research.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 11402016).

References

[1] X. Deng, M. Mao, G. Tu, H. Liu, H. Zhang, Geometric conservation law and applications to high-order finite difference schemes with stationary grids, Journal of Computational Physics 230 (4) (2011) 1100–1115.

[2] O. Friedrich, Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids, Journal of computational physics 144 (1) (1998) 194–212.

[3] B. Cockburn, C.-W. Shu, The runge–kutta discontinuous galerkin method for conservation laws v: multidimensional systems, Journal of Computational Physics 141 (2) (1998) 199–224.

[4] H. T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous galerkin methods, in: 18th AIAA Computational Fluid Dynamics Conference, 2007, p. 4079.

[5] Z. Wang, H. Gao, A unifying lifting collocation penalty formulation including the discontinuous galerkin, spectral volume/difference methods for conservation laws on mixed grids, Journal of Computational Physics 228 (21) (2009) 8161–8186.

[6] N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven, K. Sorensen, ADIGMA–A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications: Results of a Collaborative Research Project Funded by the European Union, 2006-2009, Vol. 113, Springer Science & Business Media, 2010.

[7] J. S. Hesthaven, Numerical methods for conservation laws: From analysis to algorithms, Vol. 18, SIAM, 2018.

[8] A. Burbeau, P. Sagaut, C.-H. Bruneau, A problem-independent limiter for high-order runge–kutta discontinuous galerkin methods, Journal of Computational Physics 169 (1) (2001) 111–150.

[9] L. Krivodonova, Limiters for high-order discontinuous galerkin methods, Journal of Computational Physics 226 (1) (2007) 879–896.

[10] J. Qiu, C.-W. Shu, A comparison of troubled-cell indicators for runge–kutta discontinuous galerkin methods using weighted essentially nonoscillatory limiters, SIAM Journal on Scientific Computing 27 (3) (2005) 995–1013.

[11] J. Qiu, C.-W. Shu, Runge–kutta discontinuous galerkin method using weno limiters, SIAM Journal on Scientific Computing 26 (3) (2005) 907–929.

[12] J. Zhu, J. Qiu, C.-W. Shu, M. Dumbser, Runge–kutta discontinuous galerkin method using weno limiters ii: unstructured meshes, Journal of Computational Physics 227 (9) (2008) 4330–4353.

[13] J. Qiu, C.-W. Shu, Hermite weno schemes and their application as limiters for runge–kutta discontinuous galerkin method ii: Two dimensional case, Computers & Fluids 34 (6) (2005) 642–663.

[14] J. Zhu, J. Qiu, Hermite weno schemes and their application as limiters for runge-kutta discontinuous galerkin method, iii: unstructured meshes, Journal of Scientific Computing 39 (2) (2009) 293–321.

[15] H. Luo, J. D. Baum, R. Löhner, A hermite weno-based limiter for discontinuous galerkin method on unstructured grids, Journal of Computational Physics 225 (1) (2007) 686–713.

[16] X. Zhong, C.-W. Shu, A simple weighted essentially nonoscillatory limiter for runge–kutta discontinuous galerkin methods, Journal of Computational Physics 232 (1) (2013) 397–415.

[17] J. Zhu, X. Zhong, C.-W. Shu, J. Qiu, Runge–kutta discontinuous galerkin method using a new type of weno limiters on unstructured meshes, Journal of Computational Physics 248 (2013) 200–220.
[18] A. W. Cook, W. H. Cabot, A high-wavenumber viscosity for high-resolution numerical methods, Journal of Computational Physics 195 (2) (2004) 594–601.
[19] A. W. Cook, W. H. Cabot, Hyperviscosity for shock-turbulence interactions, Journal of Computational Physics 203 (2) (2005) 379–385.
[20] B. Fiorina, S. K. Lele, An artificial nonlinear diffusivity method for supersonic reacting flows with shocks, Journal of Computational Physics 222 (1) (2007) 246–264.
[21] S. Kawai, S. K. Lele, Localized artificial diffusivity scheme for discontinuity capturing on curvilinear meshes, Journal of Computational Physics 227 (22) (2008) 9498–9526.
[22] A. Mani, J. Larsson, P. Moin, Suitability of artificial bulk viscosity for large-eddy simulation of turbulent flows with shocks, Journal of Computational Physics 228 (19) (2009) 7368–7374.
[23] S. Premasuthan, C. Liang, A. Jameson, Computation of flows with shocks using the spectral difference method with artificial viscosity, i: Basic formulation and application, Computers & Fluids 98 (2014) 111–121.
[24] D. Moro, N. C. Nguyen, J. Peraire, Dilation-based shock capturing for high-order methods, International Journal for Numerical Methods in Fluids 82 (7) (2016) 398–416.
[25] J.-L. Guermond, R. Pasquetti, B. Popov, Entropy viscosity method for nonlinear conservation laws, Journal of Computational Physics 230 (11) (2011) 4248–4267.
[26] V. Zingan, J.-L. Guermond, J. Morel, B. Popov, Implementation of the entropy viscosity method with the discontinuous galerkin method, Computer Methods in Applied Mechanics and Engineering 253 (2013) 479–490.
[27] A. Chaudhuri, G. B. Jacobs, W.-S. Don, H. Abbassi, F. Mashayek, Explicit discontinuous spectral element method with entropy generation based artificial viscosity for shocked viscous flows, Journal of Computational Physics 332 (2017) 99–117.
[28] P.-O. Persson, J. Peraire, Sub-cell shock capturing for discontinuous galerkin methods, in: 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006, p. 112.
[29] A. Klöckner, T. Warburton, J. S. Hesthaven, Viscous shock capturing in a time-explicit discontinuous galerkin method, Mathematical Modelling of Natural Phenomena 6 (3) (2011) 57–83.
[30] B. D. Tracey, K. Duraisamy, J. J. Alonso, A machine learning strategy to assist turbulence model development, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1287.
[31] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166.
[32] K. Duraisamy, P. Durbin, Transition modeling using data driven approaches, in: Proceedings of the Summer Program, 2014, p. 427.
[33] J. Ling, J. Templeton, Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty, Physics of Fluids 27 (8) (2015) 085103.
[34] S. H. Cheung, T. A. Oliver, E. E. Prudencio, S. Prudhomme, R. D. Moser, Bayesian uncertainty analysis with applications to turbulence modeling, Reliability Engineering & System Safety 96 (9) (2011) 1137–1149.
[35] M. Emory, J. Larsson, G. Iaccarino, Modeling of structural uncertainties in reynolds-averaged navier-stokes closures, Physics of Fluids 25 (11) (2013) 110822.
[36] E. Dow, Q. Wang, Quantification of structural uncertainties in the k-w turbulence model, in: 52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t, 2011, p. 1762.
[37] E. J. Parish, K. Duraisamy, A paradigm for data-driven predictive modeling using field inversion and machine learning, Journal of Computational Physics 305 (2016) 758–774.
[38] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in reynolds-averaged navier–stokes simulations: A data-driven, physics-informed bayesian approach, Journal of Computational Physics 324 (2016) 115–136.
[39] J. S. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Springer Science & Business Media, 2007.
[40] C. F. Higham, D. J. Higham, Deep learning: An introduction for applied mathematicians, arXiv preprint arXiv:1801.05894.
[41] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941.
[42] G. Cybenko, Continuous valued neural networks with two hidden layers are sufficient, Tech. rep. (1988).
[43] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems (MCSS) 2 (4) (1989) 303–314.
[44] M. S. Gashler, S. C. Ashmore, Training deep fourier neural networks to fit time-series data, in: International Conference on Intelligent Computing, Springer, 2014, pp. 48–55.
[45] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
[46] A. Kurganov, E. Tadmor, Solution of two-dimensional riemann problems for gas dynamics without riemann problem solvers, Numerical Methods for Partial Differential Equations 18 (5) (2002) 584–608.
[47] P. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, Journal of computational physics 54 (1) (1984) 115–173.